### 代码模块化

#### 1.模块化

模块是比对象或函数稍大的、用于组织代码的单元，通过模块可以将程序进行分类。

条件：（1）模块接口，通过接口调用模块。

（2）隐藏模块内部实现，不用关注细节，直接使用。

优点：（1）更易理解和维护。

（2）提高代码的可重用性。

在ES6之前，没有内置的模块，就使用模块模式，AMD和CommonJS是两个相互竞争的标准，除了语法和原理的区别之外，主要的区别是AMD的设计理念是明确基于浏览器，而CommonJS的设计是面向通用的JavaScript环境。

#### 2.模块模式

通过使用立即执行函数、对象和闭包来创建模块的方式称为模块模式。

通过使用立即执行函数，我们可以隐藏指定的模块执行细节。通过添加对象和闭包，我们可以定义模块接口，通过接口暴露模块的功能。

    // 模块模式

const MouseCounterModule = function(){

// 创建模块私有变量

      let numclicks = 0

// 创建模块私有函数

      const handleClick = () => {

        alert(++numclicks)

      }

      return{

        countClicks:() => {

          document.addEventListener('click',handleClick)

        }

      }

    }()

缺点：模块本身常常依赖其他模块的功能，扩展的模块无法共享原有模块的内部属性。

#### 3.AMD

AMD是构建客户端Web应用程序的工具之一，很容易指定模块及依赖关系，最流行的实现是RequireJS。

优点：（1）自动处理依赖，无需考虑模块引入的顺序。

（2）异步加载模块，避免阻塞。

（3）在同一个文件中可以定义多个模块。

    // 使用AMD定义模块依赖于jQuery

    define('MouseCounterModule',['jQuery'],$ => {

      let numclicks = 0

      const handleClick = () => {

        alert(++numclicks)

      }

      return{

        countClicks: () => {

          $(document).on('click',handleClick)

        }

      }

    })

AMD提供名为define的函数，他接受以下参数

（1）新创建模块的ID。

（2）当前模块依赖的模块ID。

（3）初始化模块的工厂函数。

#### 4.CommomJS

面向通用JavaScript环境，使用基于文件的模块，所以每个文件中只能定义一个模块。

通过module.exports作为模块的公共接口，只有通过module.exports对象暴露的对象或函数才可以在模块外部访问

    // commonJS

    const $ = require('jQuery')

    let numclicks = 0

    const handleClick = () => {

      alert(++numclicks)

    }

    module.exports = {

      countClicks:() => {

        $(document).on('click',handleClick)

      }

    }

//在另一个文件中引用该模块

    const MouseCounterModule = require('MouseCounterModule.js')

    MouseCounterModule.countClicks()

优点：（1）语法简单。

（2）是Node.js默认的模块格式。

缺点：不显式的支持浏览器。

#### 5.ES6模块

ES6模块结合了CommonJS与AMD的优点，语法相对简单，每个文件就是一个模块，并且支持异步模块加载。

通过关键字export导出标识符，在其他模块中通过关键字import导入标识符。

    // 在模块最后一行导出

    const ninja = 'Youshi'

    const message = 'Hello'

    function sayHiTONinja(){

      return message + ' ' + ninja

    }

    export {message, sayHiTONinja}

//

    import { message,sayHiTONinja} from 'Ninja.js'

#### 二级标题

##### 三级标题